Computation with perturbed dynamical systems

Olivier Bournez, Daniel Graça, Emmanuel Hainry

Computation with perturbed dynamical systems[☆]

Olivier Bournez[a,*], Daniel S. Graça[b,c], Emmanuel Hainry[d,e]

[a] *Ecole Polytechnique, LIX, 91128 Palaiseau Cedex, France*
[b] *CEDMES/FCT, Universidade do Algarve, C. Gambelas, 8005-139 Faro, Portugal*
[c] *SQIG/Instituto de Telecomunicações, Lisboa, Portugal*
[d] *Université de Lorraine, Nancy, France*
[e] *LORIA, Campus Scientifique, BP 239, 54506 Vandœuvre-lès-Nancy, France*

Abstract

This paper analyzes the computational power of dynamical systems robust to infinitesimal perturbations.

Previous work on the subject has delved on very specific types of systems. Here we obtain results for broader classes of dynamical systems (including those systems defined by Lipschitz/analytic functions).

In particular we show that systems robust to infinitesimal perturbations only recognize recursive languages. We also show the converse direction: every recursive language can be robustly recognized by a computable system. By other words we show that robustness is equivalent to decidability.

Keywords: Dynamical systems, reachability, robustness, computational power, verification

1. Introduction

Recently there has been a surge of interest on the field of computer aided verification. In particular, a topic that has deserved much attention is concerned with computer aided verification of hybrid/continuous systems [1]. The idea is to get an "as automatic as possible" procedure such that, having as input the description of a system, it could tell in finite time if the system satisfies a given property.

A property which has been addressed by many authors is the reachability problem, which is concerned with the long term behavior of the system. Briefly,

[☆]This work has been partially supported by the INRIA program "Équipe Associée" ComputR. D. Graça was partially supported by *Fundação para a Ciência e a Tecnologia* and EU FEDER POCTI/POCI via SQIG - Instituto de Telecomunicações through the FCT project PEst-OE/EEI/LA0008/2011.

[*]Corresponding author

Email addresses: bournez@lix.polytechnique.fr (Olivier Bournez), dgraca@ualg.pt (Daniel S. Graça), Emmanuel.Hainry@loria.fr (Emmanuel Hainry)

this problem can be stated as follows: "given as input a region A and a point x_0 determine whether the trajectory starting on x_0 will eventually reach A."

The majority of results that appear in the literature tend to classify this problem as undecidable [2], [3], [4], [5], [6], [7] (several references are given, since each one deals with a particular class of systems). The reason is simple: many systems are known to simulate Turing machines (e.g. simple classes of linear hybrid automata [8] or piecewise constant derivative systems [9]) and one can encode the Halting problem as a reachability problem. However, these simulations usually use exact real numbers, and thus infinite precision is required to get the previous undecidability results.

Due to the use of exact simulations, some authors have questioned the meaningfulness of such results, arguing that this undecidability is due to non-robustness, sensitivity to initial values of the systems, and that it never occurs in "real systems" [10]. For example, Martin Fränzle writes in [11] "Hence, on simple information-theoretic grounds, the undecidability results thus obtained can be said to be artifacts of an overly idealized formalization. However, while this implies that the particular proof pattern sketched above lacks physical interpretation, it does not yield any insight as to whether the state reachability problem for hybrid systems featuring noise is decidable or not. We conjecture that there is a variety of realistic noise models for which the problem is indeed decidable."

Several attempts were made to understand if the reachability problem becomes decidable in the presence of noise. This appears not to be clear and to deeply depend on the considered notion of noise: it was previously known that this problem was decidable for some classes of models, such as Timed Automata [12] (independent to the issue of noise). However, it was also proved in [13] that small perturbations of the trajectory still yields undecidability for timed and hybrid systems. In opposition, using a different model for noise (infinitesimal perturbations), it was shown [11] that the reachability problem is decidable for a certain model of hybrid systems. This has been extended to several models [10]. In [14] it is shown that Turing machines exposed to small stochastic noise can decide the Halting problem, since its computational power when the error converges to 0 is $\approx \Pi_2^0$. The effect of noise was also studied in the context of neural networks. In [15] it is shown that analog neural net subjected to gaussian or other common noise distributions cannot recognize arbitrary regular languages. In [16] (see [17] for a more throughout discussion) the authors show that some types of neural networks can simulate Turing machines. If arbitrary (possibly non-recursive) reals are authorized, it has been shown by same authors in [18] that any arbitrary language can be recognized by a neural network in exponential time, and that languages recognized in polynomial time correspond to non-uniform polynomial time, that is to say to languages recognized by circuits of polynomial size.

In this paper we will continue the work done in [10], in which the authors consider several classes of widely used models of dynamical systems: Turing machines, piecewise affine maps, linear hybrid automata, and piecewise constant derivative systems. They introduce a notion of "perturbed" dynamics for each

of these classes and then establish the computational power required to solve the reachability problem. Their idea is to use the reachability relation $\mathcal{R}$: given two points x and y in the state space, they are in relation (denoted by $x\mathcal{R}y$) if there is a trajectory from x to y. Then they perturb this relation in the following manner: given an $\varepsilon > 0$, $x\mathcal{R}_\varepsilon y$ if there is a ε-perturbed trajectory from x to y (the precise definition depends if the system is discrete-time or continuous-time). Taking the limit (intersection) of the relations $\mathcal{R}_\varepsilon$ when $\varepsilon \to 0$, one obtains the relation $\mathcal{R}_\omega$. They call the system robust if $\mathcal{R} = \mathcal{R}_\omega$. By other words, a system is robust if its reachability relation does not change under infinitesimal perturbations of the dynamics.

This idea of infinitesimal robustness has a close resemblance with the notion of "structural stability" for dynamical systems: a system $\mathcal{A}$ is structurally stable if, roughly, ε-perturbed systems converge to $\mathcal{A}$ as $\varepsilon \to 0$, a concept widely studied in the dynamical system theory see e.g. [19], [20].

It is proved in [10] that for Turing machines, piecewise affine maps, linear hybrid automata, and piecewise constant derivative systems, the relation R_ω belongs to the class Π^1_0 (it is co-recursively enumerable), and moreover, any Π^1_0 relation can be reduced to a relation R_ω of a perturbed system: any complement of a recursively enumerable set can be semi-decided by an infinitesimally perturbed system. This result shows that robustness implies decidability. Indeed, the reach set is recursively enumerable and if the system is robust, the complement of the reach set must be recursively enumerable, from which it follows that the reach set must be recursive for robust systems.

In short, the results of [10] give a partial answer to the above mentioned conjecture: the reachability problem is decidable for certain classes of robust systems, if the notion of robustness is the one considered in [10]. By other words, undecidability of verification arises only when non-robust systems are considered.

The aim of this paper is to extend the results of [10] for the case of Lipschitz/analytic and computable (in the sense of recursive analysis [21]) systems. This class of systems is very broad including any C^k-system for $k \geq 1$ (e.g. any system defined using the usual functions of analysis – polynomials, exponential, trigonometric functions, etc. – is Lipschitz). We present both continuous-time and discrete-time versions of our results. We therefore seek to strengthen the results of [10]: verification of the reachability problem is decidable for robust systems considered in classical mathematics and computer science. These results suggest that undecidability of verification is really a by-product of non-robustness, even if the system does not have the dynamics of Turing machines, piecewise affine maps, linear hybrid automata, and piecewise constant derivative systems as in [10].

In a more provocative way, undecidability of verification for safety properties seems to be an artifact of modelization for very general and natural classes of systems.

The present paper is an extended journal version of preliminary results presented at the conference MFCS 2010 [22]. In particular, this paper differs from the conference version on the following points: first, the proofs of the results are

fully detailed (and a minor correction is stated); second, we study both acceptance and recognition; third, results are extended for the case where the state space may be unbounded, sometimes at the price of more subtle arguments.

Since this paper deals with recognized/accepted languages in bounded/unbounded domains using discrete/continuous-time, each basic result will have several variants, depending on which hypothesis is being used. Since all these cases may yield some confusion, next we provide a small guide for the main results.

Overall, we have two main results in the paper: robustness implies recursiveness and its converse result. The results which show that robustness implies recursiveness are: Theorem 16 (bounded domain and discrete/continuous-time), Theorem 17 (unbounded domain, discrete-time), Theorem 18 (unbounded domain, continuous-time). For the converse result we have: Theorem 20 (unbounded domain, continuous-time), Theorem 21 (unbounded domain, discrete-time), Theorem 22 (bounded domain, continuous-time), and finally Theorem 23 (bounded domain, discrete-time).

Throughout the paper we use the following assumptions: (i) unperturbed systems acting as language acceptors or recognizers are deterministic; (ii) words to be inspected for language membership are encoded by initial states rather than inspected sequentially by a dynamical system.

For the result stating that recursiveness implies robustness we use a particular coding of words into initial states. Albeit being restricted to this particular coding, we conjecture that our results can be extended to more general types of encodings.

We should also note that our results for bounded domains assume non-uniform robustness: the amount of robustness depends on the initial state. This is natural since we are coding infinitely many words as initial states in a bounded space and therefore the infimum of robustness (distance between initial states) over all (permissible, i.e. word encoding) initial states must be zero.

Notice that this paper discusses continuous time and discrete time dynamical systems in a uniform way, but separately. Some of the constructions then carry to hybrid systems, that is, to systems with continuous and discrete dynamics: on the one hand, it is possible to embed discrete time dynamical systems and continuous dynamical systems in well chosen classes of hybrid systems, and on the other hand it is possible to extend some of our constructions to approximate trajectories for continuous systems to hybrid systems. For succinctness, we do not fully discuss in this paper hybrid systems and refer instead the reader to [23], [24].

2. Preliminaries

2.1. Computable analysis

In this paper we will deal with computable systems in the sense of computable analysis. This section briefly introduces some basic material from computable analysis.

The theory of computation can be rooted in the seminal work of Turing, Church, and others, which provided a framework in which to achieve computation over discrete identities or, equivalently, over the integers.

However, this definition was not enough to cover computability over continuous structures, and was then developed by other authors such as Turing himself [25], Grzegorczyk [26], or Lacombe [27] to originate *computable analysis* (also known as *recursive analysis*).

The idea underlying computable analysis to compute over a set A is to encode each element a of A by a countable sequence of "simple" elements, called a *name*. Each sequence (name) can encode at most one element of A. The more elements we have from a sequence encoding a, the more precisely we can pinpoint a. From this point of view, it suffices to work only with names when performing a computation over A. To compute with names, we use oracle Turing machines as in [28]. See [29] or [21] for other equivalent approaches to computable analysis.

In the present work we will encode a real number α by a sequence of dyadic rational number, i.e. by numbers with the format $k/2^n$, where $k, n \in \mathbb{Z}$. In turn, each dyadic rational can be encoded by three natural numbers $p, q, r \in \mathbb{N}$ satisfying

$$\frac{k}{2^n} = \frac{p-q}{2^r}$$

and these three natural numbers can be encoded in just one number via well-known polynomial-time computable bijections $\langle \cdot, \cdot, \cdot \rangle : \mathbb{N}^3 \to \mathbb{N}$ [30].

Formally (see [28]), let $\nu_{\mathbb{Q}} : \mathbb{N} \to \mathbb{Q}$ be the following representation (many other natural representations of rational numbers can be chosen: they still yield the same class of computable functions – see [21, 28]) of dyadic rational numbers by integers: $\nu_{\mathbb{Q}}(\langle p, q, r \rangle) \mapsto \frac{p-q}{2^r}$, where $\langle \cdot, \cdot, \cdot \rangle : \mathbb{N}^3 \to \mathbb{N}$ is a polynomial-time computable bijection. Let us now introduce the notion of computable real point.

Definition 1. *A sequence of integers $(x_i)_{i\in\mathbb{N}} \in \mathbb{N}^{\mathbb{N}}$ converges quickly toward x (denoted by $(x_i)_{i\in\mathbb{N}} \rightsquigarrow x$) if $|\nu_{\mathbb{Q}}(x_i) - x| < 2^{-i}$ for all $i \in \mathbb{N}$.*

Definition 2. *A point $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ is called computable if for all $j \in \{1, \ldots, d\}$, there is a computable sequence $(x_i)_{i\in\mathbb{N}} \in \mathbb{N}^{\mathbb{N}}$ (i.e. a computable function $a : \mathbb{N} \to \mathbb{N}$ such that $x_i = a(i)$ for all $i \in \mathbb{N}$) satisfying $(x_i)_{i\in\mathbb{N}} \rightsquigarrow x_j$.*

We can also define computable functions.

Definition 3. *Let $X \subseteq \mathbb{R}^d$. A function $f : X \subset\to \mathbb{R}$ is called computable if there exists some d-oracle Turing machine M such that, for all $\mathbf{x} = (x_1, \ldots, x_d) \in X$ and all sequences $(x_i^j)_{i\in\mathbb{N}} \rightsquigarrow x_j$, M computes in finite time a value x'_i such that $(x'_i)_{i\in\mathbb{N}} \rightsquigarrow f(\mathbf{x})$, provided M is given as input a value $i \in \mathbb{N}$ and as oracles the d sequences $(x_i^1)_{i\in\mathbb{N}}, \ldots, (x_i^d)_{i\in\mathbb{N}}$. A function $f : X \subset \mathbb{R}^d \to \mathbb{R}^k$ is said computable if all its projections are.*

Finally, we provide the notion of computable open and closed set (cf. [21], Definition 5.1.15).

Definition 4. 1. *An open set $E \subseteq \mathbb{R}^n$ is called recursively enumerable (r.e. for short) open if there are computable sequences $\{a_i\}_{i\in\mathbb{N}}$,$\{r_i\}_{i\in\mathbb{N}}$ of integers such that*

$$E = \cup_{k=0}^{\infty} B(\nu_{\mathbb{Q}}(a_k), \nu_{\mathbb{Q}}(r_k)).$$

where $B(a,r) = \{x \in E : \|x - a\| < r\}$.

2. *A closed subset $A \subseteq \mathbb{R}^n$ is called r.e. closed if there exist computable sequence of integers $\{b_i^j\}_{i\in\mathbb{N}}$, for $j = 1, \ldots, n$ such that $\{(\nu_{\mathbb{Q}}(b_i^1), \ldots, \nu_{\mathbb{Q}}(b_i^n))\}_{i\in\mathbb{N}}$ is dense in A. A is called co-r.e. closed if its complement A^c is r.e. open. A is called computable (or recursive) if it is both r.e. and co-r.e.*
3. *An open set $E \subseteq \mathbb{R}^n$ is called computable (or recursive) if E is r.e. open and its complement E^c is r.e. closed.*

2.2. Dynamical systems

In its essence a dynamical system is a pair consisting of a state space where the action occurs and a function f which defines the evolution of the system along time. Formally it can be defined as follows [31]. Let $\mathcal{S}$ be the space of *states* and $\mathcal{T}$ the space of times (in general $\mathcal{T}$ needs only be a monoid, although usually $\mathcal{T} = \mathbb{Z}$ – discrete-time systems – or $\mathcal{T} = \mathbb{R}$ – continuous-time systems).

Definition 5. *A* dynamical system *is a triple $(\mathcal{T}, \mathcal{S}, \phi)$, where $\mathcal{S}$ is a set, $\mathcal{T}$ is a monoid, and $\phi : \mathcal{T} \times \mathcal{S} \to \mathcal{S}$ is a function satisfying (we write $\phi_t(\mathbf{x}) = \phi(t, \mathbf{x})$ for $t \in \mathcal{T}$):*

1. *$\phi_0 : \mathcal{S} \to \mathcal{S}$ is the identity function, i.e. $\phi_0(\mathbf{x}) = \mathbf{x}$*
2. *$\phi_t \circ \phi_s = \phi_{t+s}$ for every $t, s \in \mathcal{T}$.*

In this paper we will be concerned with systems defined on a continuous state space, i.e. where $\mathcal{S} \subseteq \mathbb{R}^n$ is an open set. We will only consider discrete-time ($\mathcal{T} = \mathbb{Z}$) and continuous-time systems ($\mathcal{T} = \mathbb{R}$). In the latter case, it can be shown [31] that if ϕ is a C^1 function, then the continuous-time dynamical system can be written as a differential equation

$$\mathbf{x}' = f(\mathbf{x}).$$

For this reason we will model continuous-time dynamical systems as ordinary differential equations and just say that $(\mathcal{S}, f)$ is a continuous-time dynamical system. From Definition 5, it is also not difficult to infer that, for discrete-time dynamical systems, we can reconstruct ϕ just knowing $g = \phi_1$ since (for $t > 0$. A similar reasoning applies for the other cases)

$$\phi_t(\mathbf{x}) = \underbrace{g(g(\cdots g}_{t \text{ times}}(\mathbf{x}) \cdots).$$

For this reason we will just say that $(\mathcal{S}, g)$ is a discrete-time dynamical system. Now we can introduce the notion of trajectory.

Definition 6. *Let $\mathcal{S} \subset \mathbb{R}^d$, and consider some function $f : \mathcal{S} \to \mathcal{S}$. Then a* trajectory *associated to the dynamical system $(\mathcal{S}, f)$ is:*

- *A sequence of points $\{\mathbf{x}_0, \mathbf{x}_1, ...\} \in \mathcal{S}^{\mathbb{N}}$, satisfying $f(\mathbf{x}_{i+1}) = \mathbf{x}_i$ for all $i \in \mathbb{N}$, for discrete-time dynamical systems.*
- *A solution of the differential equation $\dot{\mathbf{x}} = f(\mathbf{x})$, $\mathbf{x}(0) = \mathbf{x}_0 \in \mathcal{S}$ for continuous-time dynamical systems*

In this paper we consider dynamical systems as recognizers of languages. Let Σ denote the alphabet $\Sigma = \{0, 1\}$ and Σ^* denote the set of words over this alphabet. To recognize a language with a dynamical system, we need to encode words of Σ^* as points of $\mathcal{S}$. Here we use a variation of the encoding defined in [32], [7]. In this latter paper a word $w = w_0 \dots w_n \in \Sigma^*$ can be encoded as an integer

$$y = w_0 + w_1 2 + \ldots + w_n 2^n. \tag{1}$$

In this paper we will consider two cases: (i) the state space is compact (bounded) and (ii) the state space is $\mathbb{R}^n$ (unbounded). For the latter case one can consider the encoding (1). For the compact case we consider, without loss of generality, that $\mathcal{S} = [-1, 1]$. We can then encode a word $w \in \Sigma^*$ in the state space $\mathcal{S}$ using the encoding $v : \Sigma^* \to \mathcal{S}$ defined by

$$v(w) = \frac{2}{\pi} \arctan(w_0 + w_1 2 + \ldots + w_n 2^n). \tag{2}$$

In order to avoid pathological behaviour which may trivially lead to non-computability, we assume that computation in a dynamical systems setting has the following property: the computation is carried out in a region $V_{compute}$ and, when the computation should finish, it diverges to regions V_{accept} and V_{reject} (depending on if the word should be accepted or rejected). We assume that the (Hausdorff) distance between these 3 regions is non-zero so that we can distinguish them in finite time. This assumption is done to avoid trivial undecidability due to the impossibility of distinguishing two reals in the recursive analysis setting.

We can now put all pieces together to obtain the following definition.

Definition 7 (Considering a dynamical system as a language recognizer). *Let Σ be an alphabet, $\mathcal{H}$ a discrete/continuous-time dynamical system over space $\mathcal{S}$. Let $V_{compute}$, V_{accept}, $V_{reject} \subseteq \mathcal{S}$ be computable sets and $\sigma : \Sigma^* \to V_{compute}$ be a map satisfying the following requirements:*

1. *The Hausdorff distance between any pair of sets from the following: $V_{compute}$, V_{accept}, V_{reject} is non-zero;*
2. *There is a connected open set A which overlaps $V_{compute}$ and V_{accept}, but such that $A \cap V_{reject} = \varnothing$ (by other words, there are paths which go from $V_{compute}$ to V_{accept}, without crossing V_{reject}).*
3. *There is a connected open set B which overlaps $V_{compute}$ and V_{reject}, but such that $B \cap V_{accept} = \varnothing$.*

4. *Once a trajectory reaches V_{accept} or V_{reject}, it stays there.*

We say that $\mathcal{H}$ accepts *a language $L \subset \Sigma^*$ (or that L is the language of $\mathcal{H}$), if the following holds: for all $w \in \Sigma^*$, $w \in L$ if the trajectory of $\mathcal{H}$ starting from $\sigma(w)$ reaches V_{accept}. If $w \notin L$, we require that the corresponding trajectory always stays in $V_{compute}$ or goes to V_{reject}.*

We say that $\mathcal{H}$ recognizes *a language $L \subset \Sigma^*$ (or that L is decided by $\mathcal{H}$), if the following holds: for all $w \in \Sigma^*$, if $w \in L$ then the trajectory of $\mathcal{H}$ starting from $\sigma(w)$ reaches V_{accept}; if $w \notin L$ then the trajectory of $\mathcal{H}$ starting from $\sigma(w)$ reaches V_{reject}.*

We finish this section by recalling the notion of Lipschitz function.

Definition 8. *A function $f : X \subseteq \mathbb{R}^m \to \mathbb{R}^k$ is said Lipschitz over a set if there is some $K > 0$ such that for all $\mathbf{x}, \mathbf{y} \in X$ one has*

$$\|f(\mathbf{x}) - f(\mathbf{y})\| \leq K \|\mathbf{x} - \mathbf{y}\|. \tag{3}$$

A function $f : E \to \mathbb{R}^m$, $E \subseteq \mathbb{R}^l$, is said to be locally Lipschitz on E if it satisfies a Lipschitz condition on every compact set $V \subseteq E$.

In particular it is well known that C^1 functions are locally Lipschitz over $\mathbb{R}^n$, which imply that they are Lipschitz over a compact set $X \subseteq \mathbb{R}^m$. In [33] a notion of effectively Lipschitz function was presented, which will be used on what follows.

Definition 9. *Let $E = \cup_{n=0}^{\infty} B(a_n, r_n) \subseteq \mathbb{R}^l$, where $\overline{B(a_n, r_n)} \subseteq E$, be a r.e. open set. A function $f : E \to \mathbb{R}^m$ is called effectively locally Lipschitz on E if there exists a computable sequence $\{K_n\}$ of positive integers such that*

$$\|f(x) - f(y)\| \leq K_n \|x - y\| \text{ whenever } x, y \in \overline{B(a_n, r_n)}.$$

In particular, it can be shown [33] that any C^1 computable function on $\mathbb{R}^l$ is effectively Lipschitz.

2.3. Robustness

Before proceeding with our results, we recall the notion of robustness from [10], based on an idea of [34]. This will be the notion of robustness used throughout this paper.

Definition 10 (ε-perturbation). *Consider a discrete/continuous-time dynamical system $\mathcal{H} = (X, f)$. Given $\varepsilon > 0$, its ε-perturbation $\mathcal{H}_\varepsilon$ is the discrete/continuous-time system $\mathcal{H}_\varepsilon$ defined over the same space X, where:*

1. *$(\mathbf{x}_0, \mathbf{x}_1, ...)$ is a (ε-perturbed) trajectory of $\mathcal{H}_\varepsilon$, in the case where $\mathcal{H}$ is discrete-time, if $\|\mathbf{x}_{i+1} - f(\mathbf{x}_i)\| \leq \varepsilon$ for all $i \in \mathbb{N}$;*
2. *$\phi : \mathbb{R}_0^+ \to X$ is a (ε-perturbed) trajectory of $\mathcal{H}_\varepsilon$, in the case where $\mathcal{H}$ is continuous-time, if $\phi(0) \in X$ and $\|\phi'(t) - f(\phi(t))\| \leq \varepsilon$ for all $t \in \mathbb{R}_0^+$.*

Notice that the system $\mathcal{H}_\varepsilon$ is not, in general, deterministic. Since we are interested in robust recognition of languages by a system, our next goal is to introduce an appropriate definition of robust recognition. Some care is needed, since it may happen that a perturbed system has trajectories which go to V_{accept} and to V_{reject}.

Definition 11. *Let $\mathcal{H}_\varepsilon$ be a perturbed system.*

1. *The language $L_\varepsilon \subseteq \Sigma^*$ accepted by the system $\mathcal{H}_\varepsilon$ is formed by all words $w \in \Sigma^*$ such that there is an ε-trajectory starting from $\sigma(w)$ which reaches V_{accept}.*
2. *The language K_ω is formed by all words $w \in \Sigma^*$ with the property that, for each $w \in K_\omega$, there is an $\varepsilon > 0$ such that all trajectories of $\mathcal{H}_\varepsilon$ with origin in $\sigma(w)$ vanish in V_{reject}.*

The language K_ω can be seen as the set of words "easy to reject".

Definition 12 (Robustness). *A dynamical system $\mathcal{H}$ is said to have robust recognition if the language L it recognizes has the following properties: (i) if $w \in L$ then $w \in L_\varepsilon$ for every $\varepsilon > 0$; (ii) if $w \notin L$ then $w \in K_\omega$.*

3. Main results

We are now ready to present the main results of our paper. Section 3.1 shows that, for bounded domains, any language which can be robustly recognized is recursive. Section 3.2 shows a similar result, but for unbounded domains. Then in Section 3.3 we show the converse result: any recursive language can be robustly recognized by some system.

3.1. Bounded domain

Before presenting the main result of this section, let us present an auxiliary result, first for discrete-time systems, and later for continuous-time systems.

Theorem 13. *Let $\mathcal{H} = ([-1,1]^d, f)$ be a discrete-time system, where f is Lipschitz and computable. Then the set K_ω is recursively enumerable.*

Proof. Let $n \in \mathbb{N}\backslash\{0\}$ and $\mathcal{S} = [-1,1]^d$. One can decompose $\mathcal{S}$ in d-dimensional hypercubes $V_1, ..., V_s$ of size $\frac{1}{n}$. Using this decomposition we build a finite automaton A_n, whose states are $V_1, ..., V_s$, that roughly recognizes $L_{\frac{1}{n}}$. To complete the description of this automaton we need to define two things: (i) the set of accepting states and (ii) the transition rule δ_n. The set of accepting states consists of those hypercubes which overlap V_{accept}. These hypercubes can easily be identified since the set V_{accept} is computable (since we are only dealing with accepted languages, there is no need to consider V_{reject}).

Now let us present the transition rule of A_n. The following construction is depicted in Fig. 1. Let V_j be some hypercube. Then pick its central point $\mathbf{x}_j$ (this is an easily computable rational) and compute a rational approximation

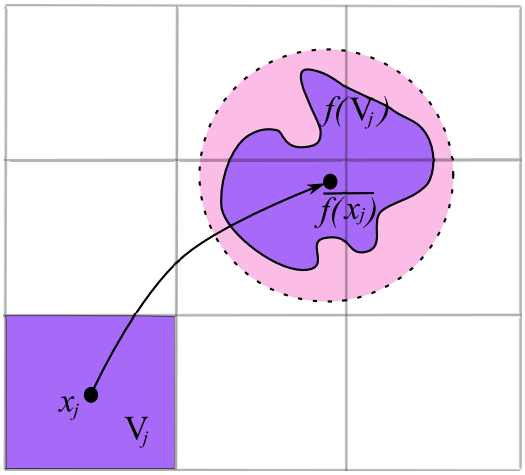

Figure 1: A figure depicting various elements used in the demonstration of Theorem 13.

$\overline{f(\mathbf{x}_j)}$ of $f(\mathbf{x}_j)$ with precision $\frac{1}{n}$. Because f is Lipschitz, there will be some Lipschitz constant $K > 0$ satisfying condition (3) for all $\mathbf{x}, \mathbf{y} \in X$. Then, if $\mathbf{x} \in V_j$ is another point of the same hypercube and $\mathbf{y}$ is an ε-perturbed image of $f(\mathbf{x})$, with $\varepsilon = 1/n$, we have

$$\|f(\mathbf{x}) - \mathbf{y}\| \leq \frac{1}{n} \quad \Rightarrow \tag{4}$$
$$\left\|\overline{f(\mathbf{x}_j)} - \mathbf{y}\right\| \leq \left\|\overline{f(\mathbf{x}_j)} - f(\mathbf{x}_j)\right\| + \|f(\mathbf{x}_j) - f(\mathbf{x})\| + \|f(\mathbf{x}) - \mathbf{y}\| \leq \frac{K+2}{n}.$$

By other words, if $\mathbf{y}$ is an ε-perturbed image of a point of V_j, then this point will be within distance $\frac{K+2}{n}$ of $\overline{f(\mathbf{x}_j)}$. We use this fact in what follows.

We use the following algorithm to show that K_ω is r.e. First compute a rational approximation $\overline{f(\mathbf{x}_j)}$ of $f(\mathbf{x}_j)$ with precision $\frac{1}{n}$. After computing $\overline{f(\mathbf{x}_j)}$, determine all the hypercubes which are within distance $\leq (K+2)/n$ of this point (in Fig. 1 this corresponds to all hypercubes covered by the ball of center $\overline{f(\mathbf{x}_j)}$ and radius $(K+2)/n$). This can be done algorithmically, in finite time, since it is only necessary to check which are the hypercubes (which are finitely many) that have vertices within distance $\leq (K+2)/n$ of $\overline{f(\mathbf{x}_j)}$. Let $W_1, ..., W_i$ be these hypercubes. Then we define the transition rule over the hypercubes as follows: $\delta(V_j) = \{W_1, ..., W_i\}$. This defines the automaton A_n.

Now we say that a point $\mathbf{x} \in X$ is accepted by A_n if it lies in an accepted hypercube. Let $\widetilde{L}_{\frac{1}{n}}$ be the language accepted by A_n. It is easy to conclude that the dynamics of A_n includes those of the $\frac{1}{n}$-perturbed system $\mathcal{H}_{\frac{1}{n}}$. Hence $L_{\frac{1}{n}} \subseteq \widetilde{L}_{\frac{1}{n}}$. On the other side, using (4), it is not difficult to see that the dynamics of A_n are included in those of $\mathcal{H}_{\frac{K+3}{n}}$ (here we suppose that $\|\mathbf{x}\| = \|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$. However, a similar result holds for other norms since

all norms are equivalent in a finite-dimensional space). Therefore

$$L_{\frac{1}{n}} \subseteq \widetilde{L}_{\frac{1}{n}} \subseteq L_{\frac{K+3}{n}} \quad \Rightarrow \quad \cap_{n=1}^{\infty} \widetilde{L}_{\frac{1}{n}} = \cap_{\varepsilon>0} L_{\varepsilon} = L_{\omega}.$$

Let us now show that K_ω is recursively enumerable. Let $w \in K_\omega$. We notice that if w belongs to K_ω, then there will be some $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, $w \notin \widetilde{L}_{\frac{1}{n}}$. Reciprocally, if $w \notin \widetilde{L}_{\frac{1}{n}}$ for some $n \in \mathbb{N}$, then $w \in K_\omega$. Moreover we can decide in finite time, using the automaton A_n, whether $w \in \widetilde{L}_{\frac{1}{n}}$. All these facts can be used to show that the following algorithm accepts words which belong to K_ω in finite time:

```
i=0
Repeat
    i++
    Simulate A_i with input ν_X(w)
Until ν_X(w) vanishes in V_reject using the dynamics of A_i
Accept w
```

This algorithms accepts in finite time exactly those words which belong to K_ω. By other words, it shows that K_ω is r.e., as required. ■

Next we present a similar result for continuous-time systems. This result can be obtained by using a construction presented in [35].

Theorem 14. *Let $\mathcal{H} = ([-1,1]^d, f)$ be a continuous-time system, where f is Lipschitz and computable. Then the set K_ω is recursively enumerable.*

Proof. The proof of this theorem is essentially the main result from [35]. There the authors show that given a differential inclusion $x' \in f(x)$ (case where f can be multivalued) one can compute an arbitrary over-approximation of the flow for this differential inclusion, with error bounded by input $\epsilon > 0$, by means of polygons. Therefore, if f is just a "normal" function (not multivalued), and $f_\epsilon(x) = [f(x) - \epsilon, f(x) + \epsilon]$, we can over-compute the flow , $x(t_0) = x_0, x' \in f_{\frac{1}{n}}(x)$ with precision $\frac{1}{n}$. Since we are on a compact, we can decide in finite time whether this polygonal over-approximation A_n vanishes entirely in V_{reject} or not. If $w \in K_\omega$, then for some over-approximation A_{n_0} (and all over-approximations A_n, with $n \geq n_0$), the trajectory starting on (the coding of) w will vanish in V_{reject}. Reciprocally, if all the trajectories starting in (the coding of) w vanish entirely in V_{reject} using the dynamics of some over-approximation A_{n_0}, then $w \in K_\omega$. Now consider the following algorithm:

```
i=0
Repeat
    i++
    Compute A_i for input ν_X(w)
Until ν_X(w) vanishes in V_reject using the dynamics of A_i
Accept w
```

This algorithm accepts in finite time exactly those words which belong to K_ω. By other words, it shows that K_ω is r.e., as required. ■

The following result is a corollary from the results of [36], [37]: one can compute any trajectory of $\mathcal{H}$. Therefore one can semi-decide if a trajectory finishes in V_{accept}, and thus semi-decide L.

Corollary 15. *Let L be the language accepted by a (continuous-time or discrete-time) system $\mathcal{H} = (\mathcal{S}, f)$, where f is Lipschitz and computable. Then L is r.e.*

Proof. We recall that the trajectory of a dynamical system $\mathcal{H}$ is computable for all times [37]. Moreover, by hypothesis, the Hausdorff distance between $V_{compute}$, V_{accept} and V_{reject} is non-zero and bounded by below by some rational $\delta > 0$. The following algorithm accepts w in finite time iff $w \in L$ ($\phi(t, \mathbf{x})$ gives the point of the trajectory starting at point $\mathbf{x}$ for $t_0 = 0$, for time t – see Definition 5):

```
i=0
Repeat
    i++
    Compute φ(i, ν_X(w)) with precision δ/2
Until φ(i, ν_X(w)) ∈ A
Accept w
```

where A is a finite over-approximation of V_{accept} (which can be computed in finite time) satisfying $dist(V_{accept}, A) < \delta/2$. If $w \notin L$, then $\phi(\mathtt{i}, \nu_X(w)) \in V_{reject} \cup V_{compute}$ for all $\mathtt{i} \in \mathbb{N}$ and the algorithm will never reach the set A and thus will run forever. If $w \in L$, there will be some $\mathtt{i} \in \mathbb{N}$ such that $\phi(\mathtt{i}, \nu_X(w)) \in V_{accept}$ and thus its $\delta/2$-estimate will reach A and stay there. Therefore this algorithm shows that L is r.e. ■

Since a set is recursive (computable) if it is r.e. as well as its complement (see for example [30]), it follows trivially that a robust system must compute recursive languages.

Theorem 16 (Robust acceptation => recursive). *Assume that language L is robustly accepted by a system $\mathcal{H} = ([-1, 1]^d, f)$, where f is Lipschitz and computable. Then L is recursive.*

Proof. L is r.e. by Corollary 15. Moreover, because the system is robust, the complement of L is K_ω, which is r.e. by Theorems 13 and 14. Hence L must be recursive. ■

3.2. Unbounded domain

We now present similar results to those of Section 3.1, but with the difference that they are valid for unbounded domains. Note that the proof of Theorem 13 is only valid for bounded domains, so it must be adapted to deal with unbounded domains. Here the hypothesis of recognition will be important.

Theorem 17 (Robust recognition => recursive I). *Let L be a language robustly recognized by a discrete-time system $\mathcal{H} = (\mathbb{R}^d, f)$, where f is effectively locally Lipschitz and computable. Then L is recursive.*

Proof. If $w \in L$ then, for ε small enough, any ε-perturbed trajectory will enter in a time T the accepting region (if $w \notin L$ these trajectories will enter in a time T the rejecting region). Then all ε-perturbed trajectories (for an input w) will stay inside some compact set A, where a Lipschitz condition holds, with some Lipschitz constant K which can be computed since f is effectively Lipschitz. Then proceeding as in the proof of Theorem 13, substituting A for $[-1,1]^d$, one gets the result. ■

A similar proof shows the following theorem.

Theorem 18 (Robust recognition => recursive II). *Let L be a language robustly recognized by a continuous-time system $\mathcal{H} = (\mathbb{R}^d, f)$, where f is effectively locally Lipschitz and computable. Then L is recursive.*

3.3. Recursiveness implies robustness

We now want to prove that any recursive language can be recognized by a robust system. By other words, we want to show the converse direction of Theorem 16 (and its unbounded variants, Theorems 17 and 18). This equals to show that every Turing machines can be simulated robustly by a (effectively) Lipschitz and computable system. We will analyze the cases in which the state space is (i) unbounded and (ii) bounded. The former case is an immediate consequence of the following theorem from [7].

Theorem 19. *Given some Turing machine M, there is an analytic and computable ODE $y' = g_M(y)$ defined over $\mathbb{R}^6$ which robustly simulates M using the encoding (1), and which is robust to perturbations $\varepsilon \leq 1/4$.*

In particular this theorem yields the following corollary, which shows the converse direction of Theorem 18:

Theorem 20 (Recursive => robust recognition). *Let M be a Turing machine. Let L be the language recognized by M. Then for the system $y' = g_M(y)$ of Theorem 19 one has $L = L_\varepsilon$ for $\varepsilon \leq 1/4$. Moreover, any ε-perturbed trajectory starting in (a coding of) $w \notin L$ will vanish in V_{reject} for $\varepsilon \leq 1/4$. In particular the complement of L is K_ω i.e. there is a computable and effectively Lipschitz system $y' = g_M(y)$ which robustly recognizes L.*

In [7], the proof of Theorem 19 is obtained by first defining a map that simulates a Turing machines robustly to errors using the encoding (1), and which is robust to perturbations $\varepsilon \leq 1/4$. Using this map instead of the differential equation, we can obtain the discrete-time counterpart of the previous theorem.

Theorem 21. *Let M be a Turing machine. Let L be the language recognized by M. Then there is a discrete-time system $\mathcal{H} = (\mathbb{R}^3, f)$, where f is computable and effectively Lipschitz, which robustly recognizes L.*

For the bounded version of this result (or, more correctly, a variant of it), we need to look at more detail how the simulation of the Turing machine M is achieved by the system $y' = g_M(y)$. Basically, the first component of this system will be used to store the left side of the tape (using the encoding (1)), the second component will be used to store the right side of the tape, and the third component is used to store the state coded as an integer. Without loss of generality, the TM can be supposed to have only two final states, one accepting and another rejecting. We can also suppose that the states are coded into integers $\{1, \ldots, m\}$, where 1 is the rejecting state, m is the accepting state, and 2 is the initial state. Once a trajectory reaches a final state (point), it stays there (modulo ε – see below).

The last 3 components of $y' = g_M(y)$ are used as memory for the main simulation. This simulation is robust to errors in the sense that for any $0 < \varepsilon < 1/4$, one can perturb any of these trajectories up to an amount ε and still be able to simulate M.

Now we use the previous construction to simulate M on a compact set $X = (-1, 1)^6$. If ϕ is a solution of $y' = g_M(y)$ simulating M on $\mathbb{R}^6$, we can pick $\phi_1 = \frac{2}{\pi} \arctan \phi$ (and hence $\phi = \tan\left(\frac{\phi_1 \pi}{2}\right)$) as the corresponding simulation of M on $(-1, 1)^6$. In general

$$\begin{aligned} \phi_1' &= (\frac{2}{\pi} \arctan \phi)' = \frac{2}{\pi} \frac{1}{1+\phi^2} \phi' = \frac{2}{\pi} \frac{1}{1+\phi^2} g_M(\phi) \implies \\ \phi_1' &= \frac{2}{\pi} \frac{1}{1+\phi^2} g_M(\phi) = \frac{2}{\pi} \frac{1}{1+\tan^2\left(\frac{\phi_1 \pi}{2}\right)} g_M\left(\tan\left(\frac{\phi_1 \pi}{2}\right)\right) = f_M(\phi_1) \end{aligned} \tag{5}$$

where

$$f_M(x) = \frac{2}{\pi} \frac{1}{1+\tan^2\left(\frac{x\pi}{2}\right)} g_M\left(\tan\left(\frac{x\pi}{2}\right)\right).$$

Hence, the system $y' = f_M(y)$ simulates M on X, with input w coded by $v(w)$, where v is given by (2). Moreover, robustness among states still exists, and the simulation of M can be carried out if the states are not perturbed more than

$$\bar{\varepsilon} = \arctan(m + \varepsilon) - \arctan(m) \tag{6}$$

We now show that any recursive language can be robustly recognized by an analytic and computable system in a bounded set. Note that since the set is open (although bounded), the function defining the system is not necessarily Lipschitz. Indeed, it is conjectured that there are recursive languages which cannot be accepted by any analytic system defined on a compact, finite dimensional space through a reasonable input and output encoding [38, Conjecture 2]. By other words, if this conjecture holds, some recursive language can only be recognized by analytic systems on bounded non-closed sets, and that's the reason why we used an open set. Of course, the system is effectively Lipschitz over its state space though the question of knowing whether a Lipschitz and computable system can robustly recognize a given recursive language remains open.

Theorem 22 (recursive => robust recognition, bounded case). *Let A be a recursive language. Then there is an analytic and computable continuous-time system over $\mathcal{S} = (-1,1)^6$ which robustly recognizes A.*

Proof. Pick $\varepsilon = 1/4$ and let M be a Turing machine which recognizes language L. Consider the construction depicted above. From all of the above, we have a dynamical system $\mathcal{H}$ defined with an ordinary differential equation (5) which allows us to robustly simulate M on the compact space $\mathcal{S}$. There will be a component of (5) (the third) which will give the state of M. The state is coded by the numbers $\arctan(1), \arctan(2), \ldots, \arctan(m)$, where $\arctan(1)$ is the rejecting state and $\arctan(m)$ is the accepting state. The idea is to pick $V_{compute}$ as the region which overlaps the states $\arctan(2), \ldots, \arctan(m-1)$ modulo the allowed error $\bar{\varepsilon}$ given by (6). V_{accept} and V_{reject} are defined similarly. Then one simulates M with a system like (5). For each trajectory simulating a computation, the trajectory will remain in $V_{compute}$ until the computation accepts or reject. At this point the trajectory goes to V_{accept} or V_{reject}, respectively and stays there.

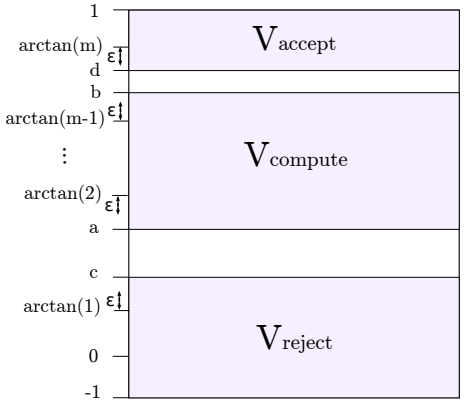

Figure 2: Computing regions for the proof of Theorem 22.

More precisely, define $V_{compute} = [a,b] \times (-1,1)^5$ where a, b are rationals satisfying

$$\arctan 1 + \bar{\varepsilon} < a \leq \arctan 2 - \bar{\varepsilon}$$
$$\arctan(m-1) + \bar{\varepsilon} \leq b < \arctan m - \bar{\varepsilon},$$

where $\bar{\varepsilon}$ is given by (6), $V_{reject} = (-1,c] \times (-1,1)^5$ where c is a rational satisfying $\arctan 1 + \bar{\varepsilon} \leq c < a$, and $V_{accept} = [d,1) \times (-1,1)^5$, where d is a rational satisfying $b < d \leq \arctan(m) - \bar{\varepsilon}$. These regions are represented in Fig. 2. Note that there is a gap between these sets, since their Hausdorff distance is $\delta = \min\{a-c, d-b\} > 0$.

Remark also that, since L is recursive, the computation of M will always halt for a given input $w \in \Sigma^*$ and therefore the contents of the tape will not

grow beyond a certain amount $size(w)$ for a given input w (bigger words will be on the region $\left[\frac{2}{\pi}\arctan size(w), 1\right)$). Therefore, since the function g_M is robust to errors, we will still be able to proceed the computation with some robustness, namely with errors less than or equal to $\gamma = \arctan(size(w) + \varepsilon) - \frac{2}{\pi}\arctan size(w)$ (the amount of error depends on the input, but this is not a problem for us. The important is that there is robustness). Therefore, for all $\delta \leq \gamma$: (i) if $w \in L$, then $w \in L_\delta$ (the δ-perturbed language of the system $\mathcal{H}$), (ii) if $w \notin L$, then all δ-perturbed trajectories vanish in V_{reject}. Therefore this system robustly recognizes L. ■

Proceeding similarly, but now using the robust map that simulates Turing machines from [7] instead of the differential equation, one obtains the following result.

Theorem 23. *Let L be a recursive language. Then there is an analytic and computable discrete-time system over $\mathcal{S} = (-1, 1)^6$ which robustly recognizes L.*

4. Conclusion

In this paper we showed that, on compact sets, robustness of dynamical systems in the sense of [10] is equivalent to decidability. It would be interesting to know what happens at a more refined level, i.e. if from a complexity point of view.

Acknowledgements. We would like to thank the anonymous referee for helpful corrections and suggestions which helped to improve the paper.

References

[1] R. Alur, G. J. Pappas (Eds.), Hybrid Systems: Computation and Control: 7th International Workshop (HSCC 2004), LNCS 2993, Springer, 2004.

[2] C. Moore, Unpredictability and undecidability in dynamical systems, Phys. Rev. Lett. 64 (20) (1990) 2354–2357.

[3] M. S. Branicky, Universal computation and other capabilities of hybrid and continuous dynamical systems, Theoret. Comput. Sci. 138 (1) (1995) 67–100.

[4] E. Asarin, O. Maler, Achilles and the tortoise climbing up the arithmetical hierarchy, J. Comput. System Sci. 57 (3) (1998) 389–398.

[5] O. Bournez, Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy, Theoret. Comput. Sci. 210 (1) (1999) 21–71.

[6] P. Koiran, C. Moore, Closed-form analytic maps in one and two dimensions can simulate universal Turing machines, Theoret. Comput. Sci. 210 (1) (1999) 217–223.

[7] D. S. Graça, M. L. Campagnolo, J. Buescu, Computability with polynomial differential equations, Adv. Appl. Math. 40 (3) (2008) 330–349.

[8] T. A. Henzinger, P. W. Kopke, A. Puri, P. Varaiya, What's decidable about hybrid automata?, J. Comput. System Sci. 57 (1) (1998) 94–124.

[9] E. Asarin, O. Maler, A. Pnueli, Reachability analysis of dynamical systems having piecewise-constant derivatives, Theoret. Comput. Sci. 138 (1995) 35–65.

[10] E. Asarin, A. Bouajjani, Perturbed Turing machines and hybrid systems, in: Logic in Computer Science, 2001. Proc. 16th Annual IEEE Symposium, 2001, pp. 269–278.

[11] M. Fränzle, Analysis of hybrid systems: An ounce of realism can save an infinity of states, in: J. Flum, M. Rodríguez-Artalejo (Eds.), Computer Science Logic (CSL'99), LNCS 1683, Springer, 1999, pp. 126–140.

[12] R. Alur, D. L. Dill, Automata for modeling real-time systems, in: Automata, Languages and Programming, 17th International Colloquium, LNCS 443, Springer, 1990, pp. 322–335.

[13] T. A.Henzinger, J.-F. Raskin, Robust undecidability of timed and hybrid systems., in: N. A. Lynch, B. H. Krogh (Eds.), Proc. Hybrid Systems: Computation and Control, Third International Workshop, HSCC 2000, LNCS 1790, Springer, 2000, pp. 145–159.

[14] E. Asarin, P. Collins, Noisy Turing machines, in: L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, M. Yung (Eds.), Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, LNCS 3580, Springer, 2005.

[15] W. Maass, E. Sontag, Analog neural nets with gaussian or other common noise distributions cannot recognize arbitrary regular languages, Neural Comp. 11 (1999) 771–782.

[16] H. T. Siegelmann, E. D. Sontag, On the computational power of neural networks, J. Comput. System Sci. 50 (1) (1995) 132–150.

[17] H. T. Siegelmann, Neural Networks and Analog Computation: Beyond the Turing Limit, Birkhäuser, 1999.

[18] H. T. Siegelmann, E. D. Sontag, Analog computation via neural networks, Theoret. Comput. Sci. 131 (2) (1994) 331–360.

[19] J. Guckenheimer, P. Holmes, Nonlinear Oscillations, Dynamical Systems, and Bifurcation of Vector Fields, Springer, 1983.

[20] M. W. Hirsch, S. Smale, R. Devaney, Differential Equations, Dynamical Systems, and an Introduction to Chaos, Academic Press, 2004.

[21] K. Weihrauch, Computable Analysis: an Introduction, Springer, 2000.

[22] O. Bournez, D. S. Graça, E. Hainry, Robust computations with dynamical systems, in: P. Hlinený, A. Kucera (Eds.), Proc. 35th International Symposium on Mathematical Foundations of Computer Science (MFCS 2010), LNCS 6281, Springer, 2010, pp. 198–208.

[23] P. Collins, J. Lygeros, Computability of finite-time reachable sets for hybrid systems, in: Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference, IEEE Computer Society Press, 2005, pp. 4688–4693.

[24] P. Collins, Semantics and computability of the evolution of hybrid systems, SIAM J. Control and Optimization 49 (2) (2011) 890–925.

[25] A. M. Turing, On computable numbers, with an application to the Entscheidungsproblem, Proc. London Math. Soc. (Ser. 2–42) (1936) 230–265.

[26] A. Grzegorczyk, On the definitions of computable real continuous functions, Fund. Math. 44 (1957) 61–71.

[27] D. Lacombe, Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles III, C. R. Acad. Sci. Paris 241 (1955) 151–153.

[28] K.-I. Ko, Computational Complexity of Real Functions, Birkhäuser, 1991.

[29] M. B. Pour-El, J. I. Richards, Computability in Analysis and Physics, Springer, 1989.

[30] P. Odifreddi, Classical Recursion Theory, Vol. 1, Elsevier, 1989.

[31] M. W. Hirsch, S. Smale, Differential Equations, Dynamical Systems, and Linear Algebra, Academic Press, 1974.

[32] D. S. Graça, M. L. Campagnolo, J. Buescu, Robust simulations of Turing machines with analytic maps and flows, in: S. B. Cooper, B. Löwe, L. Torenvliet (Eds.), CiE 2005: New Computational Paradigms, LNCS 3526, Springer, 2005, pp. 169–179.

[33] D. Graça, N. Zhong, J. Buescu, Computability, noncomputability and undecidability of maximal intervals of IVPs, Trans. Amer. Math. Soc. 361 (6) (2009) 2913–2927.

[34] A. Puri, Dynamical properties of timed automata, in: A. P. Ravn, H. Rischel (Eds.), Proc. Formal Techniques in Real-Time and Fault-Tolerant Systems, 5th International Symposium, FTRTFT'98, LNCS 1486, Springer, 1998, pp. 210–227.

[35] A. Puri, V. Borkar, P. Varaiya, Epsilon-approximation of differential inclusions, in: Proc. of the 34th IEEE Conference on Decision and Control, 1995, pp. 2892–2897.

[36] P. Collins, Continuity and computability of reachable sets, Theor. Comput. Sci. 341 (2005) 162–195.

[37] P. Collins, D. S. Graça, Effective computability of solutions of differential inclusions — the ten thousand monkeys approach, Journal of Universal Computer Science 15 (6) (2009) 1162–1185.

[38] C. Moore, Finite-dimensional analog computers: Flows, maps, and recurrent neural networks, in: C. Calude, J. Casti, M. Dinneen (Eds.), 1st International Conference on Unconventional Models of Computation - UMC'98, Springer, 1998, pp. 59–71.